(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: 2001318785 A

(43) Date of publication of application: 16.11.01

(51) Int. Cl G06F 7/72
G09C 1/00

(21) Application number: 2000137182

(22) Date of filing: 10.05.00

(71) Applicant: TOSHIBA CORP

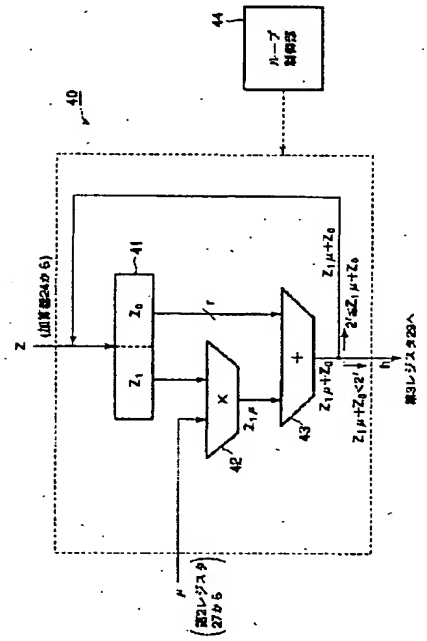
(72) Inventor: KOIKE MASANOBU
KAWAMURA SHINICHI

(54) MONTGOMERY MULTIPLICATION DEVICE AND METHOD

(57) Abstract:

PROBLEM TO BE SOLVED: To accelerate at least partial remainder operations and to accelerate the overall operation speed further by the acceleration of remainder computation.

SOLUTION: In a division circuit 40 arranged in respective product sum circuits 301-30n, though it is the remainder computation by a modulus m, the end condition of the remainder operation is mitigated so as to compare the executed result h' of the remainder computation with a value 2r larger than the modulus m. Thus, the remainder computation is accelerated by the acceleration of loop control itself by the adoption of an easily comparable number which is 2r and the reduction of the number of times of loops by the adoption of a large number which is 2r.



[What is claimed is:]

1. A Montgomery multiplication device for, when a positive integer N and positive integers x and y that are less than N are set to inputs and bases of a remainder computation system are set to $\{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_i, \ldots, b_n\}$ ($1 \leqq i \leqq n$, n is a positive integer), calculating an output w that is equal to $xyB^{-1} \bmod N$ using an integer B that is defined as multiplication of elements of the bases, comprising:

a first storage unit storing elements of the respective bases $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$ and a prior calculation result used for calculating the output w;

an approximation computation unit, when an input ξi used for expansion of the bases is received, extracting higher q bits of the input ξi, adding a present extraction value and a previous addition result excluding a highest bit and outputting a value k_i of a highest bit from the obtained addition result;

a remainder computation unit executing remainder calculation (modulus m that is equal to $2^r-\mu$ is one element of the bases, r is an integer that satisfies $m \leqq 2^r$ and μ is a nonnegative integer that is less than 2^r) while setting elements of the respective bases $\{a_1, a_2, \ldots, a_n\}$

and $\{b_1, b_2, \ldots, b_n\}$ to a modulus m based on contents of the first storage unit and the output k_i of the approximation computation unit; and

a second storage unit storing results of remainder calculation performed by the remainder computation unit, wherein

the remainder computation unit comprises:

a remainder calculation part for receiving an input z (z is a positive integer) comprising a sum (xy+d+c) of a product xy of inputs x and y inputted from the first or second storage unit, a previous remainder calculation result d and the prior calculation result c inputted based on the output k_i of the approximate computation unit, and for executing remainder calculation h= z mod m by a modulus m of the input z; and

a loop control part for comparing an execution result h' executed by the remainder calculation part with an upper limit value 2^r of the modulus m, for returning the execution result h' to the remainder calculation part as the input z when the execution result h' is equal to or greater than 2^r and for outputting the execution result h' to the second storing unit as a result h of the remainder calculation when the execution result h' is less than 2^r.

2. The Montgomery multiplication device according to claim 1, wherein

the remainder calculation part, comprises:

a bit selection part for, when executing remainder calculation h=z mod m by a modulus m of the input z, respectively selecting a value z_0 of lower r bits of the input z and a value z_1 of higher bits excluding the value z_0 of the lower bits from the input z;

a multiplication part for, when the value z_1 of higher bits extracted by the bit selection part and an input μ regarding the modulus m are received, calculating a product $z_1\mu$ of the value z_1 and the input μ; and

an addition part for calculating a sum $(z_1\mu + z_0)$ of the product $z_{1\mu}$ obtained by the multiplication part and the value z_0 of lower r bits extracted by the bit selection part, wherein

an execution result h' executed by the remainder calculation part is a sum $(z_{1\mu} + z_0)$ obtained by the addition part.

3. A Montgomery multiplication method

of, when a positive integer N and positive integers x and y that are less than N are set to inputs and bases of a remainder computation system are set to $\{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_i, \ldots, b_n\}$ ($1 \leqq i \leqq n$, n is a positive integer), calculating an output w that is equal to $xyB^{-1} \bmod N$ using an integer B that is defined as multiplication of elements of the bases, comprising:

a first storage process of storing elements of the respective bases $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$ and a prior calculation result used for calculating the output w;

an approximation computation process of, when an input ξi used for expansion of the bases is received, extracting higher q bits of the input ξi, adding a present extraction value and a previous addition result excluding a highest bit and outputting a value k_i of a highest bit from the obtained addition result;

a remainder computation process of executing remainder calculation[3] (modulus m that is equal to $2^r-\mu$ is one element of the bases, r is an integer that satisfies $m \leqq 2^r$ and μ is a nonnegative integer that is less than 2^r) while elements of the respective bases $\{a_1, a_2, \ldots, a_n\}$ and $\{b_1, b_2, \ldots, b_n\}$ are set to a modulus m based on contents of the first storage process and the output k_i of the approximation computation process; and

a second storage process of storing results of remainder calculation performed by the remainder computation process, wherein

the remainder computation process comprises:

a remainder calculation process of receiving an input z (z is a positive integer) comprising a sum (xy+d+c) of a product xy of inputs x and y inputted from storage contents of the first or second storage process, a previous remainder calculation result d and the prior calculation result c inputted based on the output k_i of the approximate computation process, and of executing remainder calculation h= z mod m by a modulus m of the input z; and

a loop control process of comparing an execution result h' executed by the remainder calculation process with an upper limit value 2^r of the modulus m, returning the execution result h' to the remainder calculation process as the input z when the execution result h' is equal to or greater than 2^r and outputting the execution result h' to the second storing process as a result h of the remainder calculation when the execution result h' is less than 2^r .

4. The Montgomery multiplication method according to claim 3, wherein

the remainder calculation process, comprises:

a bit selection process of, when executing remainder calculation $h=z \bmod m$ by a modulus m of the input z, respectively selecting a value z_0 of lower r bits of the input z and a value z_1 of higher bits excluding the value z_0 of the lower bits from the input z;

a multiplication process of, when the value z_1 of higher bits extracted by the bit selection process and an input μ regarding the modulus m are received, calculating a product $z_1\mu$ of the value z_1 and the input μ; and

an addition process of calculating a sum $(z_1\mu + z_0)$ of the product $z_1\mu$ obtained by the multiplication process and the value z_0 of the lower r bits extracted by the bit selection process, wherein

an execution result h' executed by the remainder calculation process is a sum $(z_1\mu+z_0)$ obtained by the addition process.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2001－318785
（P2001－318785A）
(43)公開日　平成13年11月16日(2001.11.16)

(51)Int.Cl.⁷	識別記号	ＦＩ	テーマコード（参考）
Ｇ０６Ｆ　7/72		Ｇ０６Ｆ　7/72	５Ｊ１０４
Ｇ０９Ｃ　1/00	６５０	Ｇ０９Ｃ　1/00	６５０Ａ　９Ａ００１

審査請求　未請求　請求項の数４　ＯＬ　（全 12 頁）

(21)出願番号　特願2000－137182(P2000－137182)

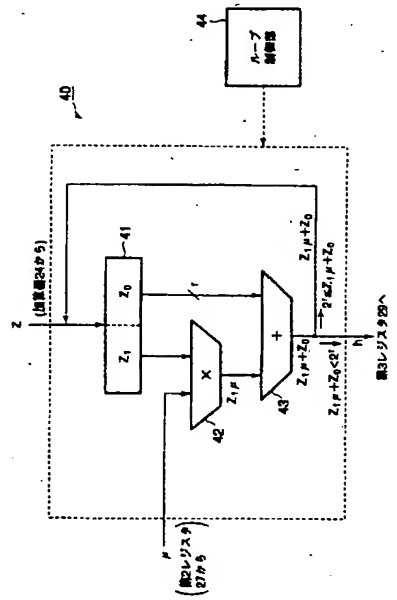
(22)出願日　平成12年５月10日(2000.5.10)

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(72)発明者　小池　正修
東京都府中市東芝町１番地　株式会社東芝府中工場内
(72)発明者　川村　信一
神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
(74)代理人　100058479
弁理士　鈴江　武彦　（外６名）

Ｆターム(参考)　5J104 AA22 JA23 NA18
9A001 BB02 EE03 GG01

(54)【発明の名称】　モンゴメリ乗算装置及び方法

(57)【要約】
【課題】　剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させる。
【解決手段】　各積和回路３０１～３０ｎに配置された除算回路４０において、法ｍでの剰余算であるにも関わらず、剰余算の実行結果ｈ’を法ｍよりも大きい値２ｒと比較するように剰余算の終了条件を緩和している。これにより、２ｒという比較の容易な数の採用によるループ制御自体の高速化と、２ｒという大きい数の採用によるループ回数の低減化とによって剰余算を高速化させる。



【特許請求の範囲】

【請求項１】　正の整数Ｎと、Ｎ未満の正の整数ｘ；ｙとを入力とし、剰余演算系の基底を｛a_1，a_2，…，a_i，…，a_n｝および｛b_1，b_2，…，b_i，…，b_n｝（$1 \leqq i \leqq n$、ｎは正の整数）としたとき、前記基底の要素の乗積として定義された整数Ｂ（$=b_1 b_2 \cdots b_n$）を用いて、出力$w = xyB^{-1} \bmod N$を算出するためのモンゴメリ乗算装置であって、

前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝と前記出力ｗの算出用の事前計算結果とが記憶される第１記憶手段と、

前記基底の拡張用の入力ξ_iを受けたとき、前記入力ξ_iの上位ｑビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値k_iを出力する近似演算手段と、

前記第１記憶手段の内容および前記近似演算手段の出力k_iに基づいて、前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝を法ｍとする剰余算（但し、法$m = 2^r - \mu$は基底の一要素、ｒはｍ≦2^rを満たす整数、μは2^r未満の非負整数）を実行する剰余演算手段と、

前記剰余演算手段による剰余算の結果が記憶される第２記憶手段とを備え、

前記剰余演算手段は、

前記第１記憶手段又は前記第２記憶手段からの入力ｘ，ｙの積ｘｙと前回の剰余算の結果ｄと前記近似演算手段の出力k_iに基づき入力される前記事前計算結果ｃとの和（ｘｙ＋ｄ＋ｃ）からなる入力ｚ（但し、ｚは正の整数）を受けると共に、前記入力ｚの法ｍによる剰余算$h = z \bmod m$を実行する剰余算部と、

前記剰余算部による実行結果ｈ’と前記法ｍの上限値2^rとを比較し、前記実行結果ｈ’が2^r以上のときには当該実行結果ｈ’を前記入力ｚとして前記剰余算部に戻し、前記実行結果ｈ’が2^r未満のときには当該実行結果ｈ’を前記剰余算の結果ｈとして前記第２記憶手段に出力するループ制御部と、

を備えたことを特徴とするモンゴメリ乗算装置。

【請求項２】　請求項１に記載のモンゴメリ乗算装置において、

前記剰余算部は、

前記入力ｚの法ｍによる剰余算$h = z \bmod m$を実行するとき、前記入力ｚの下位ｒビットの値z_0と、前記入力ｚから前記下位の値z_0を除いた上位ビットの値z_1とを夫々抽出するビット選択部と、

前記ビット選択部により抽出された上位ビットの値z_1と前記法ｍに関する入力μとを受けたとき、両者の積$z_1\mu$を算出する乗算部と、

前記乗算部により得られた積$z_1\mu$と前記ビット選択部により抽出された下位ｒビットの値z_0との和（$z_1\mu + z_0$）を算出する加算部とを備え、

前記剰余算部による実行結果ｈ’は、前記加算部により得られた和（$z_1\mu + z_0$）であることを特徴とするモンゴメリ乗算装置。

【請求項３】　正の整数Ｎと、Ｎ未満の正の整数ｘ，ｙとを入力とし、剰余演算系の基底を｛a_1，a_2，…，a_i，…，a_n｝および｛b_1，b_2，…，b_i，…，b_n｝（$1 \leqq i \leqq n$、ｎは正の整数）としたとき、前記基底の要素の乗積として定義された整数Ｂ（$=b_1 b_2 \cdots b_n$）を用いて、出力$w = xyB^{-1} \bmod N$を算出するためのモンゴメリ乗算方法であって、

前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝と前記出力ｗの算出用の事前計算結果とが記憶される第１記憶工程と、

前記基底の拡張用の入力ξ_iを受けたとき、前記入力ξ_iの上位ｑビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値k_iを出力する近似演算工程と、

前記第１記憶工程の記憶内容および前記近似演算工程の出力k_iに基づいて、前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝を法ｍとする剰余算（但し、法$m = 2^r - \mu$は基底の一要素、ｒはｍ≦2^rを満たす整数、μは2^r未満の非負整数）を実行する剰余演算工程と、

前記剰余演算工程による剰余算の結果が記憶される第２記憶工程とを含んでおり、

前記剰余演算工程は、

前記第１記憶工程の記憶内容又は前記第２記憶工程の記憶内容からの入力ｘ，ｙの積ｘｙと前回の剰余算の結果ｄと前記近似演算工程の出力k_iに基づき入力される前記事前計算結果ｃとの和（ｘｙ＋ｄ＋ｃ）からなる入力ｚ（但し、ｚは正の整数）を受けると共に、前記入力ｚの法ｍによる剰余算$h = z \bmod m$を実行する剰余算工程と、

前記剰余算工程による実行結果ｈ’と前記法ｍの上限値2^rとを比較し、前記実行結果ｈ’が2^r以上のときには当該実行結果ｈ’を前記入力ｚとして前記剰余算工程に戻し、前記実行結果ｈ’が2^r未満のときには当該実行結果ｈ’を前記剰余算の結果ｈとして前記第２記憶工程に出力するループ制御工程と、

を含んでいることを特徴とするモンゴメリ乗算方法。

【請求項４】　請求項３に記載のモンゴメリ乗算方法において、

前記剰余算工程は、

前記入力ｚの法ｍによる剰余算$h = z \bmod m$を実行するとき、前記入力ｚの下位ｒビットの値z_0と、前記入力ｚから前記下位の値z_0を除いた上位ビットの値z_1とを夫々抽出するビット選択工程と、

前記ビット選択工程により抽出された上位ビットの値ｚ

１と前記法mに関する入力μとを受けたとき、両者の積ｚ１μを算出する乗算工程と、
前記乗算工程により得られた積ｚ１μと前記ビット選択工程により抽出された下位ｒビットの値ｚ０との和（ｚ１μ＋ｚ０）を算出する加算工程とを含んでおり、
前記剰余算工程による実行結果ｈ’は、前記加算工程により得られた和（ｚ１μ＋ｚ０）であることを特徴とするモンゴメリ乗算方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、剰余演算系におけるモンゴメリ乗算アルゴリズムを用いるモンゴメリ乗算装置及び方法に係り、特に、剰余算のループ制御の高速化により、全体の処理速度を向上し得るモンゴメリ乗算装置及び方法に関する。

【０００２】

【従来の技術】近年、公開鍵暗号の重要性が増すにつれ、公開鍵暗号における大きな整数の剰余乗算に関し、その高速化を図る旨の要望が高まっている。例えば現在、事実上の標準（デファクトスタンダード）であるＲＳＡ（Rivest-Shamir-Adleman）方式では、1,024ビットの整数による剰余演算を高速に行なう必要がある。

【０００３】係る剰余演算の高速化の一方法として、剰余演算系（Residue Number System、以後ＲＮＳという）が知られている。このＲＮＳは、互いに素な比較的小さな整数の組｛ａ１，ａ２，…，ａｎ｝を用意し、演算対象の大きな整数をこれらの整数で割った余りの組で表現する方式である。

【０００４】以後、これら割る数の組｛ａ１，ａ２，…，ａｎ｝をＲＮＳの基底と呼び、基底を構成する要素の個数ｎを基底のサイズと呼ぶ。また基底の集合をａ＝｛ａ１，ａ２，…，ａｎ｝で表す。

【０００５】例えば、整数ｘと基底｛ａ１，ａ２，…，ａｎ｝が与えられたとき、ｘを基底ａｉ（ｉ＝１，２，…，ｎ）で割った余りｘｉの組（ｘ１，ｘ２，…，ｘｎ）がｘのＲＮＳ表現である。ここで、ｘｉ＝ｘ　ｍｏｄ　ａｉである。

【０００６】このとき、ｘは、基底の全要素の積Ａ＝ａ１ａ２…ａｎを法として一意的に表現できる。すなわち、ｘがＡ未満の正整数であれば、ｘとそのＲＮＳ表現（ｘ１，ｘ２，…，ｘｎ）は一対一に対応する。

【０００７】ＲＮＳ表現の数値同士の剰余乗算方法は、ポッシュら（Posch et al）による提案（K. C. Posch, R. Posch, “Modulo Reduction in Residue Number Systems”, IEEE Transaction on Parallel and Distributed Systems, Vol.6, No.5, May 1995, pp.449-454 及び “RNS-Modulo Reduction Upon a Restricted Base Value Set and its Applicability to RSA Cryptography”, Computer & Security, Vol.17,pp.637-650, 1998）並びに川村による提案（特願平１１－３１０６１９号）がある。

【０００８】これらの剰余乗算方法は、ＲＮＳ表現での演算において不利な除算を避けるために、モンゴメリ（Montgomery）の提案による剰余乗算方法（以下、モンゴメリ乗算という。詳細は P. L. Montgomery, “Modular Multiplication without Trial Division”, Mathematics of Computation, Vol.44,No.170,pp.519-521,April,1985）を利用することと、あるＲＮＳ基底で表現された整数を、基底拡張を用いて別の基底で表現すること、などが特徴として挙げられる。

【０００９】ここで、基底拡張は、例えば、整数ｘにおいて、基底ａ＝｛ａ１，ａ２，…，ａｎ｝での表現（ｘ　ｍｏｄ　ａ１，ｘ　ｍｏｄ　ａ２，…，ｘ　ｍｏｄ　ａｎ）から基底ａ∪｛ｂ１｝＝｛ａ１，ａ２，…，ａｎ，ｂ１｝での表現（ｘ　ｍｏｄ　ａ１，ｘ　ｍｏｄ　ａ２，…，ｘ　ｍｏｄ　ａｎ，ｘ　ｍｏｄ　ｂ１）を生成する操作である。

【００１０】この例では、基底のサイズを１つ増やしたが、この操作を繰り返して基底のサイズをｎ’個（ｎ’は整数）増やし、その基底の下でのＲＮＳ表現を作り出すことも基底拡張と呼ぶ。

【００１１】続いて、以上のような剰余乗算方法について具体的に説明する。図４は川村により提案されたモンゴメリ乗算装置の構成を示す模式図である。このモンゴメリ乗算装置は、ＲＮＳ表現でのモンゴメリ乗算を実行するものであり、本発明者並びに川村らによりCox-Rowerアーキテクチャと命名された構成を備えている。係るCox-Rowerアーキテクチャでは、互いに並列接続されてそれぞれ剰余乗算を行なうｎ個のRowerユニットに対し、補正項を計算する１個のCoxユニットが接続されている。なお、ｎは基底のサイズである。

【００１２】ここで、Coxユニットは、基底拡張のためのものであり、ビット選択部１１及び加算器１２からなる。ビット選択部１１は、RowerユニットのＲＡＭ２１ｉから入力されたｒビットの整数ξｉの上位ｑビットを取り出して加算器１２に与える。

【００１３】加算器１２は、そのｑビットの整数を前回の加算結果に加算し、得られた加算結果のｑ＋１ビット目（加算のキャリー１ビット）に位置する１ビットの数ｋｉを各Rowerユニットに出力する。ここでｒは基底ａの各要素ａ１～ａｎのビット数であり、ｑはｒ未満の正整数である。

【００１４】川村によれば、1,024ビットのべき乗剰余算のパラメータをｎ＝３３、ｒ＝３２、ｑ＝７と取れば良いことが分かる。従って、Coxユニットの処理は、７ビット程度の加算で済むので、高速に実行可能である。ｎ個のRowerユニットは、ＲＮＳの基底を法とした剰余演算（ｆｇ＋ｃ＋ｈ）ｍｏｄ　ｍを計算可能なものであり、ＲＮＳ基底の下での剰余演算を行なう積和回路２０１～２０ｎ、ＲＡＭ２１１～２１ｎ及びＲＯＭ２２１～

２２ｎからなる。ここで、ｍはＲＮＳ基底のある一要素であり、ｆ，ｇ，ｃ，ｄ（前回のｈ）はｍ未満の整数である。各積和回路２０ｉ（但し１≦ｉ≦ｎ）は、図５に示すように、（ｆｇ＋ｃ＋ｄ）ｍｏｄ　ｍを計算するユニットであり、乗算器２３、加算器２４、スイッチ２５、第１レジスタ２６、第２レジスタ２７、剰余演算器２８及び第３レジスタ２９を備えている。乗算器２３は、入力ｆ，ｇの積ｆｇを算出して加算器２４に出力する。加算器２４は、その出力ｆｇとスイッチ２５からの入力ｃ、及び第３レジスタ２９内の前回の剰余演算結果ｄを加算し、得られた結果（ｆｇ＋ｃ＋ｄ）を剰余演算器２８に出力する。なお、スイッチ２５は、Coxユニットからの１ビットの入力ｋｉに基づいて、ＲＯＭ２２ｉから第１レジスタ２６に格納された入力ｃを加算器２４に入力又は遮断するものである。

【００１５】剰余演算器２８は、ＲＯＭ２２ｉから第２レジスタ２７に格納された基底要素ａｉ又はｂｉとしての法ｍを用い、加算器の出力結果ｆｇ＋ｃ＋ｄを法ｍで割った剰余演算（ｆｇ＋ｃ＋ｄ）ｍｏｄ　ｍを実行する。

【００１６】ここで、ポッシュ又は川村のモンゴメリ乗算アルゴリズムによれば、ＲＮＳの基底としてｍ＝２ｒ－μ（但し、μ≪２ｒ）の形式を適用可能なため、剰余演算器における剰余算ｚ　ｍｏｄ　ｍ、（但し、ｚは整数）を次のように加算と乗算で実行可能である。

【００１７】

While（ｚ＜ｍ）｛
ｚ０←ｚ　ｍｏｄ　２ｒ
ｚ１←ｚ／２ｒの商　（すなわちｚ＝ｚ１２ｒ＋ｚ０、ｚ０＜２ｒ）
ｚ　←ｚ１μ＋ｚ０
｝

なお、このアルゴリズムは次式に基づいている。

【００１８】

ｚ＝ｚ１２ｒ＋ｚ０
＝ｚ１（２ｒ－μ＋μ）＋ｚ０
＝ｚ１（２ｒ－μ）＋ｚ１μ＋ｚ０
≡ｚ１μ＋ｚ０　ｍｏｄ　ｍ　（ｍ＝２ｒ－μのため）

係るｚ０とｚ１の計算は、剰余算（ｚ１μ＋ｚ０）ｍｏｄ　ｍがあるものの、ｚ０がｚの下位ｒビットを抽出して得られ、ｚ１がｚをｒビット右シフトして得られるため、計算機で十分高速に実行可能である。

【００１９】剰余演算器２８は、このように剰余演算ｈ＝（ｆｇ＋ｃ＋ｄ）ｍｏｄ　ｍ＝（ｚ１μ＋ｚ０）ｍｏｄ　ｍを実行する。ここで、実行中の剰余演算結果ｈ’は、図示しないループ制御部により、法ｍと比較され、法ｍよりも大きい場合もしくは同じ場合（ｍ≦ｈ’）、再度、入力側に戻されて前述した抽出・乗算・加算からなる剰余算が繰返され、法ｍよりも小さい場合（ｈ’＜ｍ）には剰余演算結果ｈ（以下、剰余ともいう）として第３レジスタ２９に出力される。なお、大小比較は、減算ｈ’－ｍにより行われる。

【００２０】第３レジスタ２９内の剰余演算結果ｈは、積和回路２０ｉから出力されてＲＡＭ２１ｉに書込まれる一方、次回の積和回路２０ｉの演算時には前回の剰余演算結果ｄとして加算器２３に与えられる。

【００２１】

【発明が解決しようとする課題】しかしながら従来のモンゴメリ乗算装置及び方法では、ポッシュ又は川村の方式において、各積和回路２０ｉの剰余演算器２８における剰余算ｍｏｄ　ｍが加算や乗算と比較して処理量が多いので、Rowerユニットの演算を高速化、ひいてはCox-Rowerアーキテクチャによる演算を高速化するためのネックとなっている。

【００２２】このため、剰余算ｍｏｄ　ｍの更なる高速化が望まれている。

【００２３】本発明は上記実情を考慮してなされたもので、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現し得るモンゴメリ乗算装置及び方法を提供することを目的とする。

【００２４】

【課題を解決するための手段】本発明の骨子は、Rowerユニットでの除算の条件を緩和し、剰余算を効率的に行なうことにあり、具体的には、Rowerユニットの積和回路における剰余算の終了条件を緩和することにある。

【００２５】例えば、従来の剰余算ｈ＝ｚ　ｍｏｄ　ｍの終了条件は、実行結果ｈがｍよりも小さいとき（ｈ＜ｍ）である。これに対し、本発明に係る剰余算ｈ＝ｚ　ｍｏｄ　ｍの終了条件は、実行結果ｈが２ｒよりも小さいとき（ｈ＜２ｒ）である。ここで、ｒはｍを２進表現したときのビット数である。すなわち、ｍ＜２ｒの関係がある。従って、本発明は、実行結果ｈが、法ｍよりも大きい値２ｒ未満に下がった時点で剰余算を終了するので、従来よりも剰余算を高速化することができる。

【００２６】また、このように終了条件を緩和しても、後述する誤差の評価により演算の正確性を確認しているので、演算の信頼性を維持することができる。さて以上のような本発明の骨子に基づいて、具体的には以下のような手段が講じられる。請求項１に対応する発明は、正の整数Ｎと、Ｎ未満の正の整数ｘ，ｙとを入力とし、剰余演算系の基底を｛ａ１，ａ２，…，ａｉ，…，ａｎ｝および｛ｂ１，ｂ２，…，ｂｉ，…，ｂｎ｝（１≦ｉ≦ｎ、ｎは正の整数）としたとき、前記基底の要素の乗積として定義された整数Ｂ（＝ｂ１ｂ２…ｂｎ）を用いて、出力ｗ＝ｘｙＢ$^{-1}$ｍｏｄ　Ｎを算出するためのモンゴメリ乗算装置であって、前記各基底の要素｛ａ１，ａ２，…，ａｎ｝および｛ｂ１，ｂ２，…，ｂｎ｝と前記出力ｗの算出用の事前計算結果とが記憶される第１記憶

手段と、前記基底の拡張用の入力ξ_iを受けたとき、前記入力ξ_iの上位ｑビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値k_iを出力する近似演算手段と、前記第１記憶手段の内容および前記近似演算手段の出力k_iに基づいて、前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝を法ｍとする剰余算（但し、法$m=2^r-\mu$は基底の一要素、ｒはｍ≦2^rを満たす整数、μは2^r未満の非負整数）を実行する剰余演算手段と、前記剰余演算手段による剰余算の結果が記憶される第２記憶手段とを備え、前記剰余演算手段としては、前記第１記憶手段又は前記第２記憶手段からの入力ｘ，ｙの積ｘｙと前回の剰余算の結果ｄと前記近似演算手段の出力k_iに基づき入力される前記事前計算結果ｃとの和（ｘｙ＋ｄ＋ｃ）からなる入力ｚ（但し、ｚは正の整数）を受けると共に、前記入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄ　ｍを実行する剰余算部と、前記剰余算部による実行結果ｈ’と前記法ｍの上限値2^rとを比較し、前記実行結果ｈ’が2^r以上のときには当該実行結果ｈ’を前記入力ｚとして前記剰余算部に戻し、前記実行結果ｈ’が2^r未満のときには当該実行結果ｈ’を前記剰余算の結果ｈとして前記第２記憶手段に出力するループ制御部と、を備えたモンゴメリ乗算装置である。

【００２７】また、請求項２に対応する発明は、請求項１に対応するモンゴメリ乗算装置において、前記剰余算部としては、前記入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄｍを実行するとき、前記入力ｚの下位ｒビットの値z_0と、前記入力ｚから前記下位の値z_0を除いた上位ビットの値z_1とを夫々抽出するビット選択部と、前記ビット選択部により抽出された上位ビットの値z_1と前記法ｍに関する入力μとを受けたとき、両者の積$z_1\mu$を算出する乗算部と、前記乗算部により得られた積$z_1\mu$と前記ビット選択部により抽出された下位ｒビットの値z_0との和（$z_1\mu+z_0$）を算出する加算部とを備え、前記剰余算部による実行結果ｈ’は、前記加算部により得られた和（$z_1\mu+z_0$）であるモンゴメリ乗算装置である。

【００２８】さらに、請求項３に対応する発明は、正の整数Ｎと、Ｎ未満の正の整数ｘ，ｙとを入力とし、剰余演算系の基底を｛a_1，a_2，…，a_i，…，a_n｝および｛b_1，b_2，…，b_i，…，b_n｝（$1\leqq i\leqq n$、ｎは正の整数）としたとき、前記基底の要素の乗積として定義された整数Ｂ（$=b_1b_2\cdots b_n$）を用いて、出力$w=xyB^{-1}\bmod N$を算出するためのモンゴメリ乗算方法であって、前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝と前記出力ｗの算出用の事前計算結果とが記憶される第１記憶工程と、前記基底の拡張用の入力ξ_iを受けたとき、前記入力ξ_iの上位ｑビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値k_iを出力する近似演算工程と、前記第１記憶工程の記憶内容および前記近似演算工程の出力k_iに基づいて、前記各基底の要素｛a_1，a_2，…，a_n｝および｛b_1，b_2，…，b_n｝を法ｍとする剰余算（但し、法$m=2^r-\mu$は基底の一要素、ｒはｍ≦2^rを満たす整数、μは2^r未満の非負整数）を実行する剰余演算工程と、前記剰余演算工程による剰余算の結果が記憶される第２記憶工程とを含んでおり、前記剰余演算工程としては、前記第１記憶工程の記憶内容又は前記第２記憶工程の記憶内容からの入力ｘ，ｙの積ｘｙと前回の剰余算の結果ｄと前記近似演算工程の出力k_iに基づき入力される前記事前計算結果ｃとの和（ｘｙ＋ｄ＋ｃ）からなる入力ｚ（但し、ｚは正の整数）を受けると共に、前記入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄ　ｍを実行する剰余算工程と、前記剰余算工程による実行結果ｈ’と前記法ｍの上限値2^rとを比較し、前記実行結果ｈ’が2^r以上のときには当該実行結果ｈ’を前記入力ｚとして前記剰余算工程に戻し、前記実行結果ｈ’が2^r未満のときには当該実行結果ｈ’を前記剰余算の結果ｈとして前記第２記憶工程に出力するループ制御工程と、を含んでいるモンゴメリ乗算方法である。

【００２９】また、請求項４に対応する発明は、請求項３に対応するモンゴメリ乗算方法において、前記剰余算工程としては、前記入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄ　ｍを実行するとき、前記入力ｚの下位ｒビットの値z_0と、前記入力ｚから前記下位の値z_0を除いた上位ビットの値z_1とを夫々抽出するビット選択工程と、前記ビット選択工程により抽出された上位ビットの値z_1と前記法ｍに関する入力μとを受けたとき、両者の積$z_1\mu$を算出する乗算工程と、前記乗算工程により得られた積$z_1\mu$と前記ビット選択工程により抽出された下位ｒビットの値z_0との和（$z_1\mu+z_0$）を算出する加算工程とを含んでおり、前記剰余算工程による実行結果ｈ’は、前記加算工程により得られた和（$z_1\mu+z_0$）であるモンゴメリ乗算方法である。

【００３０】（作用）従って、請求項１，３に対応する発明は以上のような手段を講じたことにより、剰余演算手段としては、剰余算部が、第１記憶手段又は第２記憶手段からの入力ｘ，ｙの積ｘｙと前回の剰余算の結果ｄと近似演算手段の出力k_iに基づき入力される事前計算結果ｃとの和（ｘｙ＋ｄ＋ｃ）からなる入力ｚ（但し、ｚは正の整数）を受けると共に、入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄ　ｍを実行し、ループ制御部が、剰余算部による実行結果ｈ’と法ｍの上限値2^rとを比較し、実行結果ｈ’が2^r以上のときには当該実行結果ｈ’を入力ｚとして剰余算部に戻し、実行結果ｈ’が2^r未満のときには当該実行結果ｈ’を剰余算の結果ｈとして第２記憶手段に出力する。

【００３１】このように、法ｍでの剰余算であるにも関わらず、剰余算の実行結果ｈ’を法ｍよりも大きい値２ｒと比較するように剰余算の終了条件を緩和したので、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させることができる。

【００３２】また、剰余算の終了条件を緩和したが、後述する誤差の評価により演算の正確性を確認しているので、演算結果の信頼性を維持することができる。

【００３３】また、請求項２，４に対応する発明は、剰余算部としては、ビット選択部が、入力ｚの法ｍによる剰余算ｈ＝ｚ　ｍｏｄ　ｍを実行するとき、入力ｚの下位ｒビットの値ｚ０と、入力ｚから下位の値ｚ０を除いた上位ビットの値ｚ１とを夫々抽出し、乗算部が、ビット選択部により抽出された上位ビットの値ｚ１と法ｍに関する入力μとを受けたとき、両者の積ｚ１μを算出し、加算部が、乗算部により得られた積ｚ１μとビット選択部により抽出された下位ｒビットの値ｚ０との和（ｚ１μ＋ｚ０）を算出し、剰余算部による実行結果ｈ’を、加算部により得られた和（ｚ１μ＋ｚ０）としたので、剰余算をビットの抽出・加算・乗算などの簡単な処理で実行でき、請求項１，３に対応する作用を容易且つ確実に奏することができる。

【００３４】

【発明の実施の形態】以下、本発明の一実施形態について図面を参照して説明する。始めに、本発明のモンゴメリ乗算装置及び方法に適用されるモンゴメリ乗算アルゴリズムについて説明する。まず、ＲＮＳ表現におけるモンゴメリ乗算アルゴリズムを説明する。ここで、モンゴメリ乗算の法をＮとし、入力をｘ，ｙ＜２Ｎとする。出力は、ｗ＝ｘｙＢ$^{-1}$ｍｏｄ　Ｎ、又はｗ＝ｘｙＢ$^{-1}$ｍｏｄ　Ｎ＋Ｎである。係るモンゴメリ乗算アルゴリズムは、次のような７つのステップＳＴ１～ＳＴ７で表せる。

ＳＴ１．＜ｓ＞$_a$←＜ｘ＞$_a$＜ｙ＞$_a$、＜ｓ＞$_b$←＜ｘ＊＞$_b$＜ｙ＞$_b$

ＳＴ２．＜ｔ＞$_b$←＜ｓ＞$_b$＜（－Ｎ）$^{-1}$＞$_b$

ＳＴ３．＜ｔ＞$_b$から基底拡張により＜ｔ＞$_a$∪$_b$を求める。

ＳＴ４．＜ｕ＞$_a$←＜ｔ＞$_a$＜Ｎ＞$_a$

ＳＴ５．＜ｖ＞$_a$←＜ｓ＞$_a$＋＜ｕ＞$_a$

ＳＴ６．＜ｗ＞$_a$←＜ｖ＞$_a$＜Ｂ$^{-1}$＞$_a$

ＳＴ７．＜ｗ＞$_a$から基底拡張により＜ｗ＞$_a$∪$_b$を求める。

【００３５】ここで、＜ｓ＞$_a$は、基底ａの下でのｓのＲＮＳ表現を表す。その他の＜ｔ＞$_b$なども同様である。例えば＜ｓ＞$_a$は、Ａ＝ａ１ａ２…ａｎを法とする剰余環の元ｓを基底ａの各要素ａ１～ａｎで割った余りの組（ｓ１，ｓ２，…，ｓｎ）を表す。

【００３６】基底ｂ＝｛ｂ１，ｂ２，…，ｂｎ｝は基底ａとは別のＲＮＳ基底であり、Ｂ＝ｂ１ｂ２…ｂｎとしたとき、ＡとＢの最大公約数ｇｃｄ（Ａ，Ｂ）＝１である基底である。ステップＳＴ１～ＳＴ７の正確な計算には、少なくともＮ＜Ａ，Ｎ＜Ｂという条件が必要である。この条件からｘやｙは、基底ａのみ、あるいは基底ｂのみで一意的に表現できるので、入力ｘを＜ｘ＞$_a$、＜ｘ＞$_b$のペアで表すことは冗長である。しかし、入力ｘ，ｙは、両者の積ｓのとる範囲が０≦ｓ＜Ｎ^{2}であるため、基底ａ，ｂの合併集合ａ∪ｂを基底ａ∪ｂとして始めて正しく表現される。

【００３７】次に、以上のようなモンゴメリ乗算アルゴリズムのうち、ステップＳＴ３，ＳＴ７の基底拡張について説明する。いまβをＡ未満の非負整数とし（０≦β＜Ａ）、非負整数βのＲＮＳ表現を＜β＞$_a$＝（β１，β２，…，βｎ）とする。ここで、＜β＞$_a$から基底拡張により＜β＞$_a$∪$_b$を求めるとする。

【００３８】このとき、周知の中国剰余定理から次の（１）式が成り立つ。

【００３９】

【数１】

$$\beta=\sum_{i=1}^{n}\beta_i\times(A_i^{-1}\bmod a_i)\times A_i \bmod A \quad \cdots(1)$$

【００４０】但し、Ａ$_i$＝Ａ／ａ$_i$であり、Ａ$_i^{-1}$は法ａ$_i$におけるＡ$_i$の乗法逆元である。このとき（１）式のｍｏｄＡを次の（２）式のように減算でも表せる。

※【００４１】

【数２】

※

$$\beta=\sum_{i=1}^{n}\beta_i\times(A_i^{-1}\bmod a_i)\times A_i-kA \quad \cdots(2)$$

【００４２】（２）式のように、ｍｏｄ　Ａを減算で表すとき、－Ａの倍数となる非負整数ｋがただ一つ存在する。この－Ａの倍数ｋの計算が、ポッシュや川村のＲＮＳ表現でのモンゴメリ乗算の処理速度を決めるポイントとなっている。

【００４３】次に、倍数ｋを算出する際に、効率の良い川村のアルゴリズムを説明する。（２）式の両辺をＡで割ると、次の（３）式になる。

【００４４】

【数３】

$$\beta/A=\sum_{i=1}^{n}(\beta i\times Ai^{-1}\bmod ai)/ai-k \quad \cdots(3)$$

（3）式を変形すると次の（4）式のようになる。

$$k=\sum_{i=1}^{n}(\beta i\times Ai^{-1}\bmod ai)/ai-\beta/A \quad \cdots(4)$$

（4）式から$0\leqq\beta/A<1$を用いて次の（5）式の関係を表せる。

$$k\leqq\sum_{i=1}^{n}(\beta i\times Ai^{-1}\bmod ai)/ai<k+1 \quad \cdots(5)$$

【００４５】ここで、小数部の切捨て操作を記号［］で表すと、（5）式に基づき、次の（6）式のように倍数ｋを表すことができる。

$$k=[\sum_{i=1}^{n}(\beta i\times Ai^{-1}\bmod ai)/ai] \quad \cdots(6)$$

【００４６】
【数４】

【００４７】この（6）式は、除算を含むため、川村は次の（7）式のように近似計算をして倍数ｋ’を求めている。

【００４８】
【数５】

$$k'=[\sum_{i=1}^{n}\mathrm{trunc}(\xi i)/2^{r}] \quad \cdots(7)$$

【００４９】ここで、$\xi_i=\beta_i\times A_i^{-1}\bmod a_i$である。trunc ()は、ｒビットの変数$\xi_i$のうち、上位ｑビットを維持して上位ｑビット目より下位の（ｒ－ｑ）ビットを全て０にする切捨て関数である。ｒは基底要素のビット数であり、ｑはｒ未満の正整数である。

【００５０】実際に（7）式は、ハードウェアで計算されるとき、以下のように逐次的に実行される。

※
$$\varepsilon_a=\max\{(2^r-a_i)/2^r\}$$
$$\delta_a=\max\{(\xi_i-\mathrm{trunc}(\xi_i))/a_i\} \quad \cdots(9)$$

なお、max {}はｉを１，２，…，ｎと動かしたときの最大値を表す。川村によれば、これら評価変数ε_a，δ_aを用い、入力βが次の（１０）式の範囲内にあるとき、近似計算の（7）式は正しいｋの値を与える。

【００５３】
$$n(\varepsilon_a+\delta_a)A\leqq\beta<A \quad \cdots(10)$$

また、川村によれば、入力βが次の（１１）式の範囲内のとき、近似計算の（7）式は正しいｋ又はｋ－１の値を与える。

★

※for（ｉ＝０；ｉ＜＝ｎ；ｉ＋＋）｛
$\sigma_i=\sigma_{i-1}+\mathrm{trunc}(\xi_i)/2^r$
$k_i=[\sigma_i]$
$\sigma_i=\sigma_i-k_i$
｝
ここで、$\sigma_0=\alpha$、$0\leqq\alpha<1$を初期値とする。αは誤差の補正のために導入されている。また、規則正しい処理を行なう観点から、ビット数ｒは全ての基底要素で共通とするのが望ましく、ここではそのように仮定する。

【００５１】（7）式で近似計算した倍数ｋ’は、（6）式で得られる正確な倍数ｋに対して近似誤差が生じる。そこで、（7）式の近似誤差の評価のため、次の（8）～（9）式に示すように、（7）式の分母の評価変数ε_a，と、分子の評価変数δ_aとが導入される。

【００５２】
$$\varepsilon_a=\max\{(2^r-a_i)/2^r\} \quad \cdots(8)$$

★【００５４】
$$0\leqq\beta<n(\varepsilon_a+\delta_a)A \quad \cdots(11)$$

また、（7）式に基づいて、近似値ｋ’を計算し、次の（１２）式を各ｉ＝１，２，…，ｎについて実行することで、基底ａから基底ｂの表現を作り出すことができる。但しこの（１２）式の場合は誤差が生じる可能性がある。

【００５５】
【数６】

$$\beta\bmod bi=\sum_{i-1}^{n}(\xi i\times Ai\bmod bi+ki(bi-A\bmod bi))\bmod bi \quad \cdots(12)$$

【００５６】以上の説明では、基底ａから基底ｂへの拡張方法を述べたが、記号ａと記号ｂとの交換により、基底ｂから基底ａへの基底拡張方法、および評価変数ε_b，δ_bが得られる。

【００５７】さらに、川村によれば、この近似値k'を使った基底拡張アルゴリズム（１２）式を用いて、ステップＳＴ１～ＳＴ７のＲＮＳ表現でのモンゴメリ乗算アルゴリズムを実行した場合、次の５つの条件（１３）～（１７）式を満たすときには、$w=xyB^{-1} \bmod N$，$w<2N$なる$<w>_{a \cup b}$を出力することが示されている。ここで、$\Delta=n(\varepsilon+\delta)$、$\varepsilon=\max\{\varepsilon_a, \varepsilon_b\}$、$\delta=\max\{\delta_a, \delta_b\}$である。

【００５８】$gcd(A, B)=1$　　…（１３）

$gcd(B, N)=1$　　…（１４）

$0 \leqq \Delta \leqq \alpha < 1$　　…（１５）

$4N/(1-\Delta) \leqq B$　　…（１６）

$2N/(1-\alpha) \leqq A$　　…（１７）

以上が本発明に適用されるモンゴメリ乗算アルゴリズムである。次に、以上のようなモンゴメリ乗算アルゴリズムを適用対象とする本発明の一実施形態について図面を参照しながら説明する。

【００５９】図１は本発明の一実施形態に係るモンゴメリ乗算装置の構成を示す模式図であり、図２はこのモンゴメリ乗算装置内の積和回路の構成を示す模式図であり、図３はこの積和回路内の除算回路の構成を示す模式図であって、前述した図面と同一部分には同一符号を付してその詳しい説明を省略し、ここでは異なる部分について主に述べる。

【００６０】すなわち、本実施形態は、前述したCox-Rowerユニットの高速化を図るものであり、具体的には図１～図３に示すように、前述した剰余演算器２８を有する各積和回路２０$_1$～２０$_n$に代えて、除算回路４０を有する積和回路３０$_1$～３０$_n$を備えている。

【００６１】ここで、各積和回路３０$_i$（但し$1 \leqq i \leqq n$）は、図２に示すように、$(fg+c+d) \bmod m$を計算するユニットであり、前述した剰余演算器２８に代えて、除算回路４０を備えている。なお、mは、前述同様にＲＮＳ基底のある一要素である。但し、f，g，c，dは、従来のm未満の整数とは異なり、2^r以下の正整数である。

【００６２】除算回路（剰余演算器）４０は、加算器２４の出力結果$fg+c+d$を法mで割った剰余hを計算するものであり、ハードウェア又はソフトウェアのいずれでもよいが、図３に示すように、ビット選択部４１、乗算器４２、加算器４３及びループ制御部４４を備えている。

【００６３】ビット選択部４１は、入力された整数zを下位rビットの値z_0と、入力された整数zから下位rビットを除いた上位ビットの値z_1とを夫々抽出する機能を持っている。ここで、値z_0，z_1は、夫々次式を満たすものとなる。

【００６４】$z_0=z \bmod m$

$z_1=[z/2^r]$

乗算器４２は、ビット選択部４１により抽出された上位ビットの値z_1と、法mを$m=2^r-\mu$と表したときのμとの積$z_1\mu$を計算し、得られた積$z_1\mu$を加算器４３に出力する機能をもっている。

【００６５】加算器４３は、ビット選択部４１で選択されたz_0と乗算器４２の出力$z_1\mu$を加算する。

【００６６】ループ制御部４４は、加算器４３の出力結果$z_1\mu+z_0$を法mの上限値2^rと比較し、大きい場合もしくは同じ場合（$2^r \leqq z_1\mu+z_0$（$=h'$））は$z_1\mu+z_0$を改めてzとしてビット選択部４１に入力して処理を繰返し、小さい場合は$z_1\mu+z_0$を剰余算結果hとして第３レジスタ２９を介してＲＡＭ２１$_i$に出力するように制御する機能をもっている。すなわち、除算回路４０の出力hは、$h=z \bmod m$、$h<2^r$を満たすものである。

【００６７】次に、以上のように構成されたモンゴメリ乗算装置によるモンゴメリ乗算方法について説明する。

【００６８】いま、本実施形態装置は、入力x，y、法N、基底a，bを準備し、ＲＯＭ２１$_1$～２１$_n$に各基底の要素$\{a_1, a_2, \cdots, a_n\}$および$\{b_1, b_2, \cdots, b_n\}$と出力wの算出用の事前計算結果とを記憶させた後、ステップＳＴ１～ＳＴ７のモンゴメリ乗算アルゴリズムを実行する。また、ステップＳＴ３，ＳＴ７の基底拡張は、川村のアルゴリズム（１２）式に従って実行される。このとき、Coxユニットとしては、基底の拡張用の入力ξ_iを受けると、入力ξ_iの上位qビットを抽出すると共に、この今回の抽出値と前回の最上位ビットを除く加算結果とを加算し、得られた加算結果のうち、最上位ビットの値k_iを各Rowerユニットの積和回路３０$_1$～３０$_n$に出力する。

【００６９】さて、ここでモンゴメリ乗算アルゴリズムＳＴ１～ＳＴ７内の各基底成分の計算について詳述する。例えば、ＳＴ１の計算$<s>_a \leftarrow <x>_a \cdot <y>_a$の第一成分$x_1 \cdot y_1 \bmod a_1$を考える。

【００７０】この計算は、図１中、第１のRowerユニットで行なわれ、剰余乗算が第１の積和回路３０$_1$により実行される。具体的には図２に示すように、まず積和回路３０$_1$に入力されたx_1，y_1が乗算器２３に入力される。乗算器２３はその積$x_1 \cdot y_1$を計算して加算器２４に出力する。

【００７１】この計算では、スイッチ２５が開いているので入力cが無く、前回の演算結果dも０である。このため、積$x_1 \cdot y_1$とd＝０とが加算器２４に入力される。

【００７２】加算器２４はこの和を計算して除算回路４０に出力する。この結果$x_1 \cdot y_1$と、基底a_1を$a_1=2^r-\mu_1$としたときのμ_1とが除算回路４０に入力される。

【００７３】ここで、除算回路４０は、$h=x_1 \cdot y_1 \bmod a_1$（但しh＜$2^r$）を出力する。なお、この除算回路４０は、法$a_1$の除算を実行するものの、従

来とは異なり、大小比較の対象を２ｒとして除算結果ｈを出力する。なお、除算回路４０は、大小比較の対象を２ｒとしたので、図３に示した構成に限らず、剰余算を高速化することができる。但し、ここでは図３に示した構成を例に挙げて述べる。

【００７４】まず、整数ｚ＝ｘ１・ｙ１が除算回路４０に入力される。

【００７５】除算回路４０においては、ビット選択部４１が、下位ｒビットの値ｚ０＝ｚｍｏｄ　２ｒと、残りの上位ビットの値ｚ１＝［ｚ／２ｒ］とを抽出し、上位ビットの値ｚ１と入力μ１とが乗算器４２に入力される。

【００７６】乗算器４２は積ｚ１μ１を計算して加算器４３に出力する。この出力ｚ１μ１と、ビット選択部４１で抽出された下位ビットの値ｚ０とが加算器４３に入力される。

【００７７】加算器４３は、これらの和ｈ＝ｚ１μ１＋ｚ０を計算して出力する。

【００７８】ループ制御部４４は、加算器４３からの出力ｈ’を法ｍの上限値２ｒと大小比較し、出力ｈ’が２ｒ以上のとき、出力ｈ’を改めてビット選択部の入力ｚとして処理を繰り返す。

【００７９】また、ループ制御部４４は、大小比較の結果、出力ｈが２ｒ未満のときにはループ終了と判断し、出力ｈ’を剰余ｈとして出力する。なお、本来、法ａ１の剰余算を行なうものであるが、出力ｈが法ａ１より大でもよい。

【００８０】このときｈ＝ｘ１・ｙ１　ｍｏｄ　ａ１、ｈ＜２ｒを満たす。ｈは、積和回路３０１から出力され、ＲＡＭ２１１に格納される。

【００８１】この部分以外にもRowerユニットでの剰余演算が必要なときは同様の処理を行なう。ここで、本発明の除算回路４０による剰余算の高速化について補足的に説明する。従来は、図示しないループ制御部にて、剰＊余演算結果ｈ’と、基底の一要素（例、ａ１）とを大小比較するため、減算ｈ’－ａｊを行ない、その減算結果が負になるか否かを確認することが必要である。

【００８２】一方、本発明は、加算器４３からの出力ｈ’と、基底要素のビット数ｒで示せる上限値２ｒとを大小比較するため、出力ｈ’が（ｒ＋１）ビット以上であるか否かを確認すればよいので、従来の減算よりも、効率的かつ高速にループの制御を実行できる。

【００８３】本発明は、このようなループ制御自体の高速化に加え、ループ回数自体も減少していることにより高速化を実現している。例えば、本願発明でのループ回数は、出力ｈが上限値２ｒ未満（ｈ＜２ｒ）に減少するまでの繰返し回数である。一方、従来のループ回数は、出力ｄが法ａ１未満（ｄ＜ａ１）に減少するまでの繰返し回数である。ここで、基底の要素ａ１はそのビット数ｒにおける上限値２ｒよりも小さい（ａ１＜２ｒ）。

【００８４】従って、一般に、出力ｈが上限値２ｒ未満に低減するまでのループ回数の方が、出力ｄが法ａ１未満に低減するまでのループ回数よりも少ない。すなわち、本願発明のループ回数の方が従来のループ回数よりも通常少ない。

【００８５】すなわち、本発明は、このようにループ回数が少ないことにもより、剰余算の高速化を実現することができる。（誤差の評価）本発明のモンゴメリ乗算装置及び方法を用いてステップＳＴ１～ＳＴ７のモンゴメリ乗算アルゴリズムを実行するときの誤差の評価は次の各式で与えられる。

【００８６】いま、分母の評価関数εを、ε＜２$^{-\tau}$（但しτ＞０）の範囲とする。ここで、τは、εの範囲を規定するために導入した正整数である。

【００８７】このとき、βが次の（１０ａ）式の範囲内にあれば、（７）式の近似計算は正しいｋの値を与える。

【００８８】

$$n(\varepsilon\cdot 2^{\tau}/(2^{\tau}-1)+\delta)A\leqq\beta<1 \quad \cdots(10a)$$

※【００８９】

また、βが次の（１１ａ）式の範囲内にあれば、（７）式の近似計算は正しいｋ又はｋ－１を与える。※

$$0\leqq\beta<n(\varepsilon\cdot 2^{\tau}/(2^{\tau}-1)+\delta)A \quad \cdots(11a)$$

これら（１０ａ），（１１ａ）式は、それぞれ前述した（１０），（１１）式に対応した条件であり、考えている状況も（１０），（１１）式の状況と同じとする。また、モンゴメリ乗算も前述した５つの条件（１３）～（１７）式のうち、補正項αの範囲を定めた（１５）式★

$$0\leqq n(\varepsilon\cdot 2^{\tau}/(2^{\tau}-1)+\delta)A\leqq\alpha<1 \quad \cdots(15a)$$

実用上、Ｎが1,024ビットの場合を考えると、ｎ＝３３、α＝0.5と取れば良く、このとき上のτは、τ≧２２程度と取れることが確認できる。

【００９１】このとき２$^{\tau}$／（２$^{\tau}$－１）は１に十分近く、（１０ａ），（１１ａ），（１５ａ）式の条件は、それぞれ（１０），（１１），（１５）式と対応しているが、これらとほぼ同じ条件となっている。

★を次の（１５ａ）式と置換することで、ｗ＝ｘｙＢ$^{-1}$ｍｏｄ　Ｎ、またはｗ＝ｘｙＢ$^{-1}$ｍｏｄ　Ｎ＋Ｎを出力する。

【００９０】

【００９２】すなわち、本実施形態を実行した場合であっても、誤差を正確に評価できるので、本実施形態の装置及び方法の信頼性を確保することができる。上述したように本実施形態によれば、法ｍでの剰余算であるにも関わらず、剰余算の実行結果ｈ’を法ｍよりも大きい値

２ｒと比較するように剰余算の終了条件を緩和したので、２ｒという比較の容易な数の採用によるループ制御自体の高速化と、２ｒという大きい数の採用によるループ回数の低減化とによる剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現させることができる。

【００９３】また、剰余算の終了条件を緩和したが、（１０ａ），（１１ａ），（１５ａ）式からなる誤差の評価により演算の正確性を確認しているので、演算結果の信頼性を維持することができる。

【００９４】また、除算回路４０が、ビット選択部４１、乗算部４２、加算部４３を有し、実行結果ｈ’を、加算部により得られた和（ｚ１μ＋ｚ０）とする場合、剰余算をビットの抽出・加算・乗算などの簡単な処理で実行できるので、前述した効果を容易且つ確実に奏することができる。

【００９５】また、本実施形態に係るCox-Rowerアーキテクチャは、例えばチップを搭載したボード等により実現でき、このボードをサーバ装置にさし込むことにより、サーバ上に実現させることもできる。

【００９６】なお、上記実施形態に記載した手法は、コンピュータに実行させることのできるプログラムとして、磁気ディスク（フロッピー（登録商標）ディスク、ハードディスクなど）、光ディスク（ＣＤ－ＲＯＭ、ＤＶＤなど）、光磁気ディスク（ＭＯ）、半導体メモリなどの記憶媒体に格納して頒布することもできる。

【００９７】なお、本願発明は、上記各実施形態に限定されるものでなく、実施段階ではその要旨を逸脱しない範囲で種々に変形することが可能である。また、各実施形態は可能な限り適宜組み合わせて実施してもよく、その場合、組み合わされた効果が得られる。さらに、上記各実施形態には種々の段階の発明が含まれており、開示される複数の構成用件における適宜な組み合わせにより種々の発明が抽出され得る。例えば実施形態に示される全構成要件から幾つかの構成要件が省略されることで発明が抽出された場合には、その抽出された発明を実施する場合には省略部分が周知慣用技術で適宜補われるものである。

【００９８】その他、本発明はその要旨を逸脱しない範囲で種々変形して実施できる。

【００９９】

【発明の効果】以上説明したように本発明によれば、剰余算の高速化により、少なくとも部分的な剰余演算を高速化でき、さらに全体的な演算速度の高速化を実現し得るモンゴメリ乗算装置及び方法を提供できる。

【図面の簡単な説明】

【図１】本発明の一実施形態に係るモンゴメリ乗算装置の構成を示す模式図

【図２】同実施形態における積和回路の構成を示す模式図

【図３】同実施形態における除算回路の構成を示す模式図

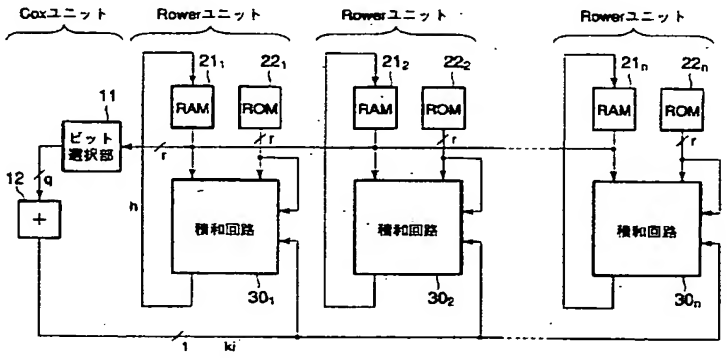
【図４】従来のモンゴメリ乗算装置の構成を示す模式図

【図５】従来の積和回路の構成を示す模式図

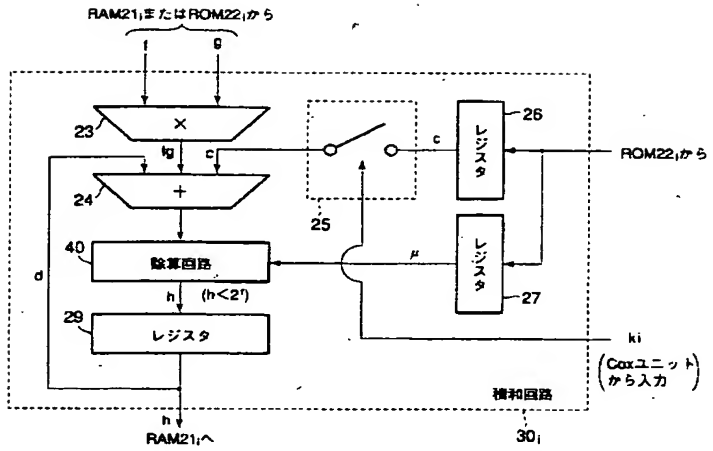
【符号の説明】

１１，４１…ビット選択部
１２，２４，４３…加算器
２３，４２…乗算器
２５…スイッチ
２６，２７，２９…レジスタ
３０１～３０ｎ…積和回路
４０…除算回路
４４…ループ制御部

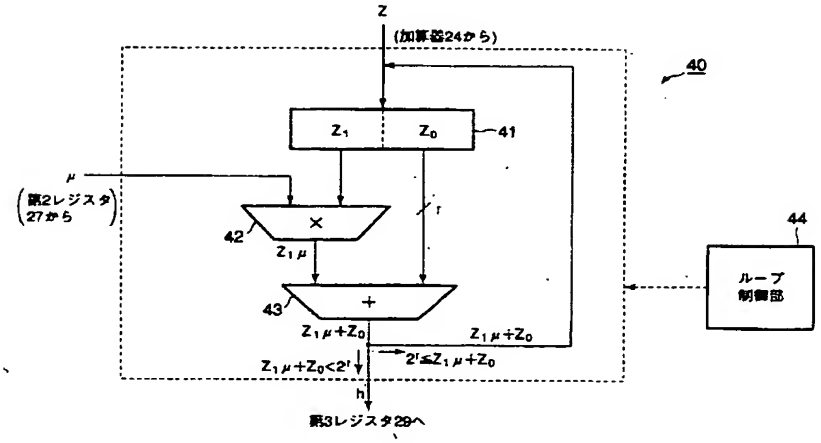
【図１】



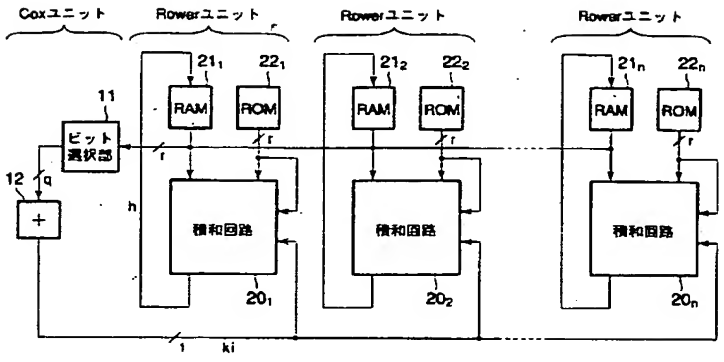
【図２】


RAM21iまたはROM22iから
f
g
23
×
fg
c
24
+
25
c
レジスタ
26
ROM22iから
40
除算回路
μ
レジスタ
27
d
h
(h<2r)
29
レジスタ
ki
(Coxユニットから入力)
積和回路
h
RAM21iへ
30i

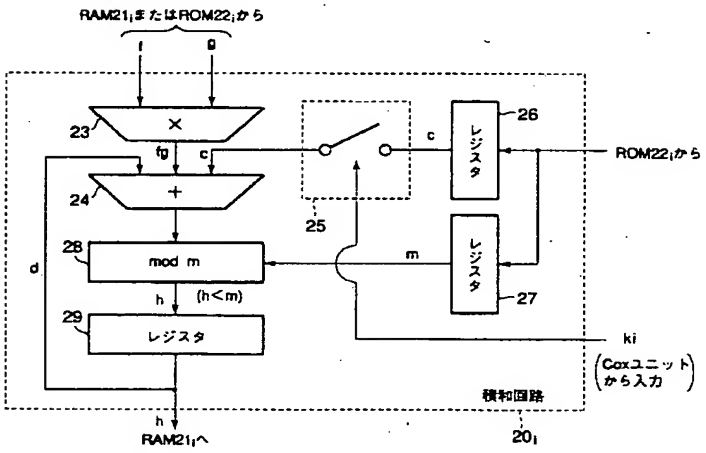
【図３】


Z
(加算器24から)
40
Z1
Z0
41
μ
(第2レジスタ27から)
42
×
r
Z1μ
43
+
44
ループ制御部
Z1μ+Z0
Z1μ+Z0
2r≦Z1μ+Z0
Z1μ+Z0<2r
h
第3レジスタ29へ

【図４】


Coxユニット
Rowerユニット
Rowerユニット
Rowerユニット
11
ビット選択部
12
q
+
h
21₁
22₁
RAM
ROM
r
積和回路
20₁
21₂
22₂
20₂
21n
22n
20n
1
ki

【図５】


RAM21iまたはROM22iから
f
g
23
×
fg
c
24
+
25
26
レジスタ
ROM22iから
28
mod m
d
m
h
(h<m)
29
レジスタ
27
ki
(Coxユニットから入力)
積和回路
20i
RAM21iへ